格式: 题号题目 + 答案 +by - “NAME”

eg: 1: What are the main features of OOPs?
Inheritance, Encapsulation, Polymorphism, Abstraction
By - Lin

PLEASE KEEP THE QUESTION NUMBER IN ASCENDING ORDER(eg: 1 1 2 3 3)

Java

42. Abstract class vs. Interface. When to use abstract class and when to use interface?

Abstract class is for creating a class without implementing all the methods since derived classes can have their own implementation. => for similar objects, not support multiple inheritance

Interface is for creating some common behavior/features that can be shared by multiple classes. => support multiple implementation, could be inherited by totally different objects

43. Java 8 features. Give an example of how you use them in your project

- forEach() method in Iterable interface.
- default and static methods in Interfaces.
- Functional Interfaces and Lambda Expressions.
- Java Stream API for Bulk Data Operations on Collections.
- Java Time API.
- Collection API improvements.
- Concurrency API improvements.
- Java IO improvements.

44. Comparable vs. Comparator
There is only one comparable method, compareto in each class. But we can create as many as comparator instances for comparing different properties.

CompareTo method will be invoked when sorted if no comparator is given. If we want to sort by different property values, we can create according property comparator and use them in Collections.sort(array, comparator).

45. Exceptions and how do you deal with exceptions in your project?

use try catch finally to deal with exceptions. specific logic for each different exception.

47Java 4 principles, and explain each of them
Polymorphism:
static => compile time => overload
dynamic => runtime => override

Encapsulation: encapsulate codes in a unit and grant control by using modifiers. Certain level of security, outside does not need to know the detailed inside processing.

Inheritance: java only support single inheritance. derived class inherit all properties and methods from parent classes.

Abstraction: use simple things to represent underlying data, like use abstraction to create classes with methods and properties, so that it can be reused for many instances.

50 Java Generic, example and advantages and how do you use generics in your project?

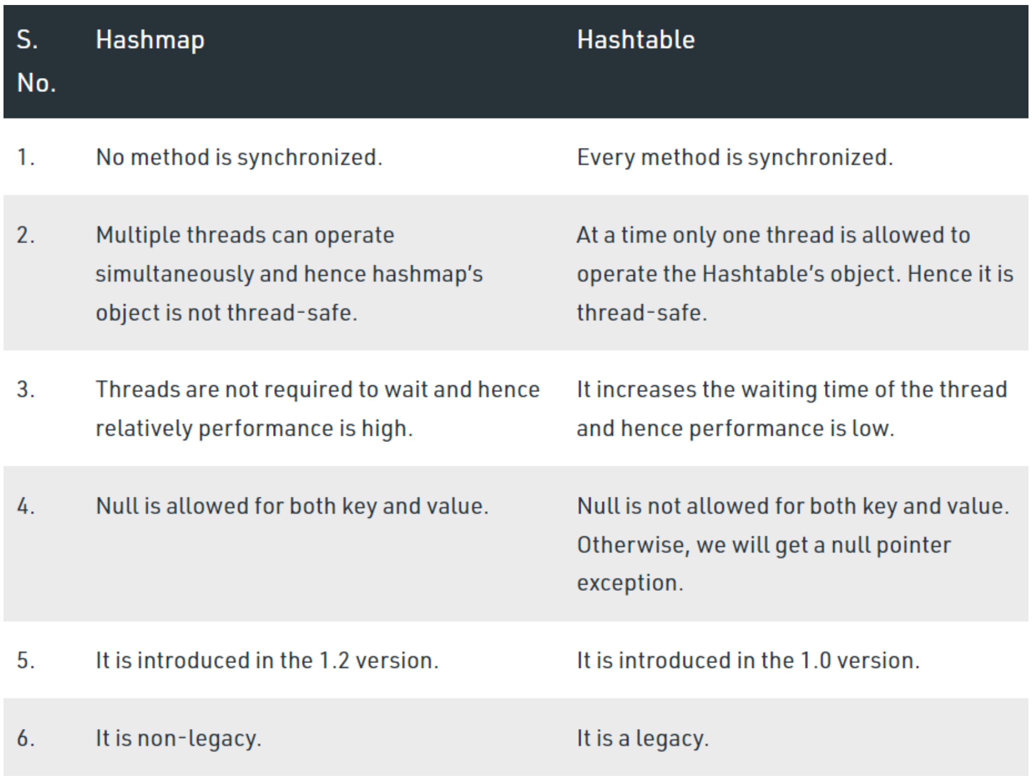
Generics makes it possible to push the runtime type check to compile time type check. It is type safe and can defer the specification of types when it is called.

Pro:
Type safe
Runtime error to compile time error
Less object type convert, so much faster

51 Serialization, explanation and example: Serialization in Java allows us to convert an Object to stream that we can send over the network or save it as file or store in DB for later usage. Examples and Deserialization can be found in
https://www.geeksforgeeks.org/serialization-in-java/
-By Lin

52 HashMap and what are the dfference with HashTable:
HashMap and Hashtable store key and value pairs in a hash table. When using a Hashtable or HashMap, we specify an object that is used as a key and the value that you want to be linked to that key. The key is then hashed, and the resulting hash code is used as the index

at which the value is stored within the table.



S. No.	Hashmap	Hashtable
1.	No method is synchronized.	Every method is synchronized.
2.	Multiple threads can operate simultaneously and hence hashmap's object is not thread-safe.	At a time only one thread is allowed to operate the Hashtable's object. Hence it is thread-safe.
3.	Threads are not required to wait and hence relatively performance is high.	It increases the waiting time of the thread and hence performance is low.
4.	Null is allowed for both key and value.	Null is not allowed for both key and value. Otherwise, we will get a null pointer exception.
5.	It is introduced in the 1.2 version.	It is introduced in the 1.0 version.
6.	It is non-legacy.	It is a legacy.

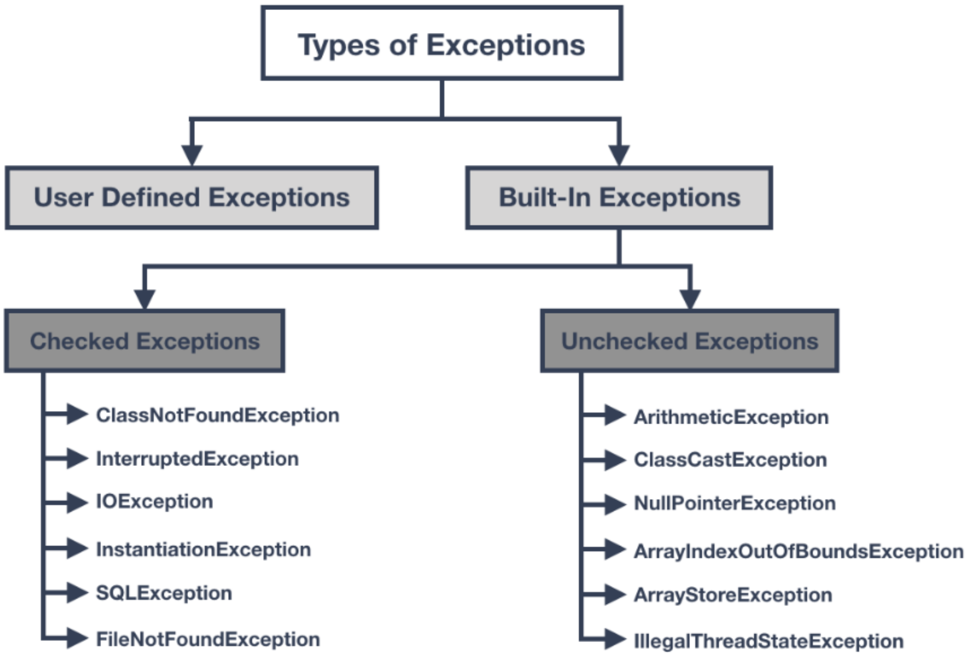
https://www.geeksforgeeks.org/differences-between-hashmap-and-hashtable-in-java/
-By Lin

53 Hashcode() and equals() / Why need to implement the hashcode() when implementing equals()?

key.hashCode() -> determine the entry index of the key
key.equals() -> determine whether two keys are the same key
(Because Hash Collision Control -> Separate Chaining)
If one.equals(two), it must have that one.hashCode() == two.hashCode()
But if one.hashCode() == two.hashCode(), it MAY have one.equals(two)
If you want to override equals(), you need also override hashCode()
- By Lin

55 Checked / unchecked exception

A checked exception in Java represents a predictable, erroneous situation, which can be checked at compile time.
eg: IOException
An unchecked exception represents an error in programming logic, not an erroneous situation, and it can be checked at runtime.
eg: ArithmeticException



- By Lin

57 How did you use lambda expression?

Lambda expressions can be stored in variables if the variable's type is an interface which has only one method. The lambda expression should have the same number of parameters and the same return type as that method.
Example shows in https://www.javatpoint.com/java-lambda-expressions
- By Lin

58 What is the Functional Interface?

A functional interface is an interface that contains only one abstract method. They can have only one functionality to exhibit.
https://www.geeksforgeeks.org/functional-interfaces-java/
- By Lin

60. Error vs. Exception

They are both superclasses of throwable class.

S.No	Errors	Exceptions
1.	The error indicates trouble that primarily occurs due to the scarcity of system resources.	The exceptions are the issues that can appear at runtime and compile time.
2.	It is not possible to recover from an error.	It is possible to recover from an exception.
3.	In java, all the errors are unchecked.	In java, the exceptions can be both checked and unchecked.
4.	The system in which the program is running is responsible for errors.	The code of the program is accountable for exceptions.
5.	They are described in the java.lang.Error package.	They are described in java.lang.Exception package

62. What happens if hash collision occurs?
It is a situation where two or more key objects produce the same final hash value and hence point to the same bucket location or array index.
64. Have you ever used any cache in your project? Explain how it works

66. Implement a singleton
creating a regular class and making sure it has:

- A private constructor
- A static field containing its only instance
- A static factory method for obtaining the instance

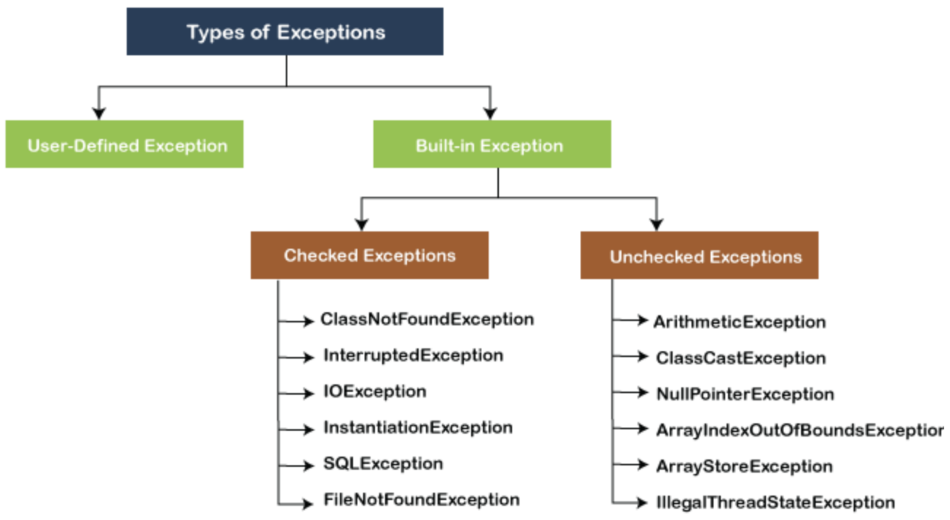
67. Do you know about the Future?
Future , represents the result of an asynchronous computation. When the asynchronous task is created, a Java Future object is returned. This Future object functions as a handle to the result of the asynchronous task
68. What design patterns did you worked on before?

70. How to custom an Exception?

- CustomException class is the custom exception class this class is extending Exception class.
- Create one local variable message to store the exception message locally in the class object.
- We are passing a string argument to the constructor of the custom exception object. The constructor set the argument string to the private string message.
- toString() method is used to print out the exception message.
- We are simply throwing a CustomException using one try-catch block in the main method and observe how the string is passed while creating a custom exception. Inside the catch block, we are printing out the message.

71. What situation will cause OutOfMemory? / How to avoid OutOfMemory?
OutOfMemoryError usually means that you're doing something wrong, either holding onto objects too long or trying to process too much data at a time.

72. Types of Exceptions



73. OutOfMemory vs. StackOverflow error

- OutOfMemoryError is related to Heap.
- StackOverflowError is related to stack

When you start JVM you define how much RAM it can use for processing. JVM divides this into certain memory locations for its processing purpose, two of those are **Stack & Heap**

If you have large objects (or) referenced objects in memory, then you will see OutofMemoryError. If you have strong references to objects, then GC can't clean the memory space allocated for that object. When JVM tries to allocate memory for new object and not enough space available it throws OutofMemoryError because it can't allocate the required amount of memory.

74. Different types of method reference

- Static methods.
- Instance methods of particular objects.
- Instance methods of an arbitrary object of a particular type.
- Constructor.

75. Talk about the stream api, lambda, optional interface and default method

76. What is functional programming? -> Compile or not

Functional programming is a paradigm that allows programming using expressions i.e. declaring functions, passing functions as arguments and using functions as statements. In Java 8 , the functional programming is supported by lambda expressions.

77.What is Optional?

to reduce the need for excessive null value checking. An Optional is a kind of wrapper object which may contain a value or not, with a few helper methods to handle existing or non-existent values

78.Internal data structure of HashMap
HashMap contains an array of the nodes, and the node is represented as a class. It uses an array and LinkedList data structure internally for storing Key and Value. There are four fields in HashMap.

79.New features of Java 8

- Interface Default and Static Methods
- Optional<T>
- Method reference
- Lambda expressions
- Repeating Annotations
- Improved type inference
- Method parameter reflection

80.Deep copy vs. shallow copy
In Shallow copy, a copy of the original object is stored and only the reference address is finally copied. In Deep copy, the copy of the original object and the repetitive copies both are stored.

81 = 44

82. Talk about multithreading you used in your project

90-97

90. How does Java code compile and run? -> JVM
At first, Java compiler (Javac, that is in Development tools), convert Java classes (with .java extension) to Bytecode. Bytecode is a file that its extension is .class. The Bytecode has supported in all operating systems (like Mac, Windows, and Linux).
To run code, we need JVM. JVM in each OS is different and to running Java, We require an JVM compatible with the operating system.
Javac (Java Compiler)pass .class file to JVM. Then, In JVM, three stage performed:

■ Class Loader
■ Bytecode Verified
■ Java-In-Time Compiler
    - by Kaisong

91. Intermediate operation and terminal operation

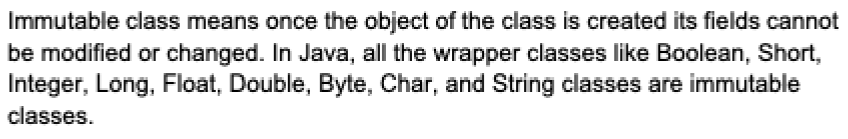
The distinction between this operations is that an intermediate operation is lazy while a terminal operation is not. When you invoke an intermediate operation on a stream, the operation is not executed immediately. It is executed only when a terminal operation is invoked on that stream. In a way, an intermediate operation is memorized and is recalled as soon as a terminal operation is invoked. You can chain multiple intermediate operations and none of them will do anything until you invoke a terminal operation. At that time, all of the intermediate operations that you invoked earlier will be invoked along with the terminal operation.

- by Kaisong

93. Java 8 callable, future

94. How do you create a thread? and How to implement multi-threads
96. Serializable and SerialVersionUID
97. How many ways to traverse a list?

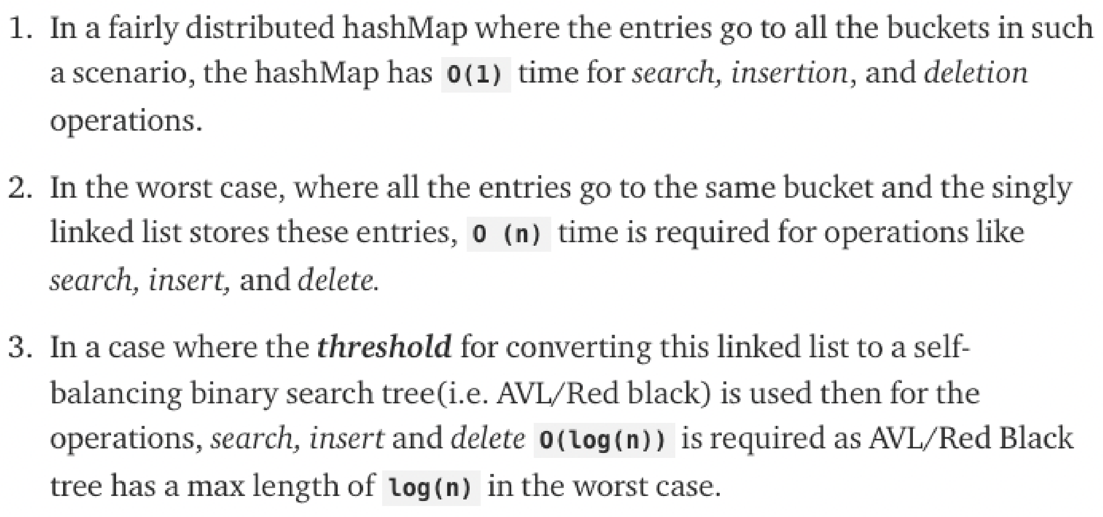
99.List some immutable objects in Java



Immutable class means once the object of the class is created its fields cannot be modified or changed. In Java, all the wrapper classes like Boolean, Short, Integer, Long, Float, Double, Byte, Char, and String classes are immutable classes.

-By Meiling Gao

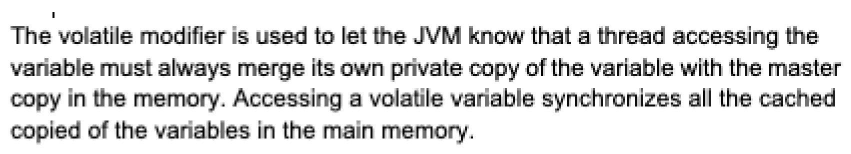
101.Time complexity of HashMap



1. In a fairly distributed hashMap where the entries go to all the buckets in such a scenario, the hashMap has `O(1)` time for *search, insertion*, and *deletion* operations.
2. In the worst case, where all the entries go to the same bucket and the singly linked list stores these entries, `O (n)` time is required for operations like *search, insert,* and *delete.*
3. In a case where the ***threshold*** for converting this linked list to a self-balancing binary search tree(i.e. AVL/Red black) is used then for the operations, *search, insert* and *delete* `O(log(n))` is required as AVL/Red Black tree has a max length of `log(n)` in the worst case.

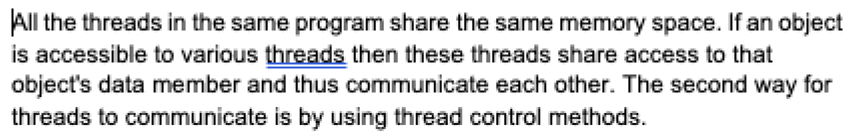
-By Meiling Gao

102.What is volatile?


The volatile modifier is used to let the JVM know that a thread accessing the variable must always merge its own private copy of the variable with the master copy in the memory. Accessing a volatile variable synchronizes all the cached copied of the variables in the main memory.

-By Meiling Gao

103.How does thread communicate with each other?


All the threads in the same program share the same memory space. If an object is accessible to various threads then these threads share access to that object's data member and thus communicate each other. The second way for threads to communicate is by using thread control methods.

-by Meiling Gao

104. What is HashMap and how to add an entry to it?

HashMap in Java is a collection based on Map and consists of key-value pairs. A HashMap is, in addition to ArrayList, one of the most widely used of Java's pre-built data structure. The HashMap is used whenever data is stored as key-value pairs, where values can be added, retrieved, deleted using keys.

Add items to it, use the put() method:

```
HashMap<String, String> capitalCities = new HashMap<String, String>();
capitalCities.put("USA", "Washington DC");
System.out.println(capitalCities);
```

-By Shilian Hou

105. thread join()

When we invoke the join() method on a thread, the calling thread goes into a waiting state. It remains in a waiting state until the referenced thread terminates.

The join() method of thread class waits for a thread to die. It is used when you want one thread to wait for completion of another. The process is like a relay race where the second runner waits until the first runner comes and hand over the flag to him.

```
public final void join()throws InterruptedException

public void join(long millis)throwsInterruptedException

public final void join(long millis, int nanos)throws InterruptedException
```

-By Shilian Hou

107. What is the advantage of Java 8 compared to Java 7?

Java 8 is a major update to the programming language which introduced a significant upgrade to the functional programming called the Lambda Expressions. Java 8 also gets a new and improved class-loader architecture, enhanced Managed Beans, multiple exceptions handling support, etc.

Major features of Java 8 include:

-Language-level support for Lambda Expressions
-Interface default and Static Methods
-Unsigned Integer Arithmetic
-Concurrent API enhancements
-New Date and Time API
-Parallel Sorting
-Null Reference Template
-New JavaScript Engine, Nashorn
-New and improved Stream API
-Removal of permanent generation
-By Shilian Hou

108. Default method vs. static method
Default methods enable you to add new functionality to the interfaces of your libraries and ensure binary compatibility with code written for older versions of those interfaces.
A static method is a method that is associated with the class in which it is defined rather than with any object.

Key	Static Interface Method	Default Method
Basic	It is a static method which belongs to the interfaces only. We can write implementation of this method in interface itself.	It is a method with default keyword and class can override this method.
Method Invocation	Static method can invoke only on interface class not on class.	It can be invoked on interface as well as class.
Method Name	Interface and implementing class, both can have static method with the same name without overriding each other.	We can override the default method in implementing class.
Use Case	It can be used as a utility method.	It can be used to provide common functionality in all implementing classes.

-By Shilian Hou

109. What is A-synchronized in multi-threading?
Synchronization in java is the capability to control the access of multiple threads to any shared resource. In the Multithreading concept, multiple threads try to access the shared resources at a time to produce inconsistent results. The synchronization is necessary for reliable communication between threads.
-By Shilian Hou

110. What is the completable future?
A compltableFuture is used for asynchronous programming. Asynchronous programming means writing non-blocking code. It runs a task on a separate thread than the main application thread and notifies the main thread about its progress, completion or failure.
In this way, the main thread does not block or wait for the completion of the task. Other tasks execute in parallel. Parallelism improves the performance of the program.
A completableFuture is a class in Java. It belongs to java.util.cocurrent package. It implements CompletionStage and Future interface.
-By Shilian Hou

124. does the front-end communicate with the back-end?

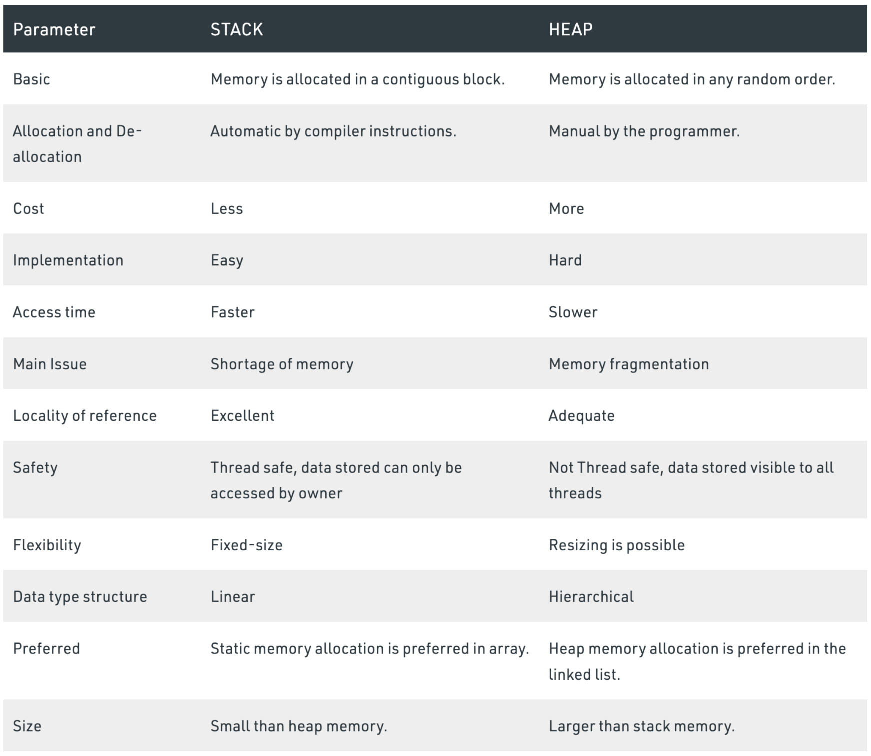
Frontend and backend communicate with each other - via Http requests.
[Frontend vs Backend (academind.com)]
-by peizhi

125.Explain heap and stack

heap: A Heap is a special Tree-based data structure in which the tree is a complete binary tree.
Max-Heap: In a Max-Heap the key present at the root node must be greatest among the keys present at all of it's children. The same property must be recursively true for all sub-trees in that Binary Tree.
Min-Heap: In a Min-Heap the key present at the root node must be minimum among the keys present at all of it's children. The same property must be recursively true for all sub-trees in that Binary Tree.

stack: Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).

compare:

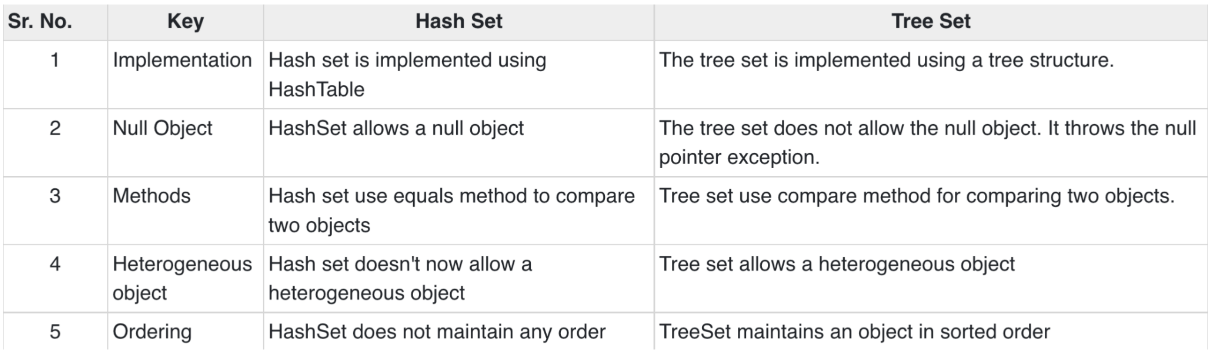


Parameter	STACK	HEAP
Basic	Memory is allocated in a contiguous block.	Memory is allocated in any random order.
Allocation and De-allocation	Automatic by compiler instructions.	Manual by the programmer.
Cost	Less	More
Implementation	Easy	Hard
Access time	Faster	Slower
Main Issue	Shortage of memory	Memory fragmentation
Locality of reference	Excellent	Adequate
Safety	Thread safe, data stored can only be accessed by owner	Not Thread safe, data stored visible to all threads
Flexibility	Fixed-size	Resizing is possible
Data type structure	Linear	Hierarchical
Preferred	Static memory allocation is preferred in array.	Heap memory allocation is preferred in the linked list.
Size	Small than heap memory.	Larger than stack memory.

-by peizhi

126.Difference between HashSet and TreeSet

Hash set and tree set both belong to the collection framework. HashSet is the implementation of the Set interface whereas Tree set implements sorted set. Tree set is backed by TreeMap while HashSet is backed by a hashmap.



Sr. No.	Key	Hash Set	Tree Set
1	Implementation	Hash set is implemented using HashTable	The tree set is implemented using a tree structure.
2	Null Object	HashSet allows a null object	The tree set does not allow the null object. It throws the null pointer exception.
3	Methods	Hash set use equals method to compare two objects	Tree set use compare method for comparing two objects.
4	Heterogeneous object	Hash set doesn't now allow a heterogeneous object	Tree set allows a heterogeneous object
5	Ordering	HashSet does not maintain any order	TreeSet maintains an object in sorted order

-by peizhi

127.How does the TreeSet sort Employee object?

How to Sort a TreeSet with User Defined Objects in Java? - GeeksforGeeks

128.Talk about factory design pattern

Factory Method is a creational design pattern that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created.

example：工厂模式 | 菜鸟教程 (runoob.com)

-by peizhi

129.Talk about observer design pattern

Observer pattern is used when there is one-to-many relationship between objects such as if one object is modified, its depenedent objects are to be notified automatically. Observer pattern falls under behavioral pattern category.

example：Design Patterns - Observer Pattern (tutorialspoint.com)

-by peizhi

Spring:

169: How does the RestTemplate work?

Rest Template is **used to create applications that consume RESTful Web Services**. You can use the exchange() method to consume the web services for all HTTP methods. The code given below shows how to create Bean for Rest Template to auto wiring the Rest Template object.

(RestTemplate是Spring Web模块提供的一个基于Rest规范提供Http请求的工具。应用中如果需要访问第三方提供的Rest接口，使用RestTemplate操作将非常方便)
by-Lin

170: How do you deal with the response get from REST api? / How to convert JSON -> Java Object?

JSON -> Java Object
The ObjectMapper class of the Jackson API provides methods to convert the Java object to JSON format or object.
The methods *writeValueAsString* and *writeValueAsBytes* of *ObjectMapper* class generate a JSON from a Java object and return the generated JSON as a string or as a byte array.
The method *readValue()* function accepts JSON String or a file containing JSON string to a Java object
https://www.baeldung.com/jackson-object-mapper-tutorial
by-Lin

171: Spring Annotation, did you use in your project?

Spring Annotations allows us to configure dependencies and implement dependency injection through java programs.
https://www.digitalocean.com/community/tutorials/spring-annotations

How do annotations work in Spring?

Spring could use its own classloader to load required classes. At runtime, when the class is loaded and Spring determines it has some appropriate annotation, **it injects bytecode to add additional properties or behavior to the class**.

-By Lin

172. What is IoC?
https://www.cnblogs.com/wmyskxz/p/8824597.html
Spring IoC is the mechanism to achieve loose-coupling between Objects dependencies. To achieve loose coupling and dynamic binding of the objects at runtime, objects dependencies are injected by other assembler objects.

Inversion of Control is a principle in software engineering which transfers the control of objects or portions of a program to a container or framework.

173. What is AOP and how do you implement AOP in the project?

What is AOP:

https://www.jianshu.com/p/007bd6e1ba1b

Spring AOP enables Aspect-Oriented Programming in spring applications. In AOP, aspects enable the modularization of concerns such as transaction management, logging or security that cut across multiple types and objects (often termed crosscutting concerns)

Way of implementing AOP:

174. Http status code you ever met

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

1. **Informational responses (100–199)**
2. **Successful responses (200–299)**
3. **Redirection messages (300–399)**
4. **Client error responses (400–499)**
5. **Server error responses (500–599)**

175. What would you do if an API response time is 500ms

1. **Cache Requests.**
2. **Prevent Abuse.**
3. **Use PATCH.**
4. **Limit Payloads.**
5. **Faster Network.**

176. Spring bean annotation, bean life cycle and bean scope

https://blog.csdn.net/qq_44543508/article/details/103718958

By definition, a Spring bean is an object that form the backbone of your application and that is managed by the Spring IoC container. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container. Otherwise, a bean is simply one of many objects in your application

Bean life cycle is managed by the spring container. When we run the program then, first of all, the spring container gets started. After that, the container creates the instance of a bean as per the request, and then dependencies are injected. And finally, the bean is destroyed when the spring container is closed.

Spring Bean Scopes allows us to have more granular control of the bean instances creation. Sometimes we want to create bean instance as singleton but in some other cases we might want it to be created on every request or once in a session.
https://developer.aliyun.com/article/619467
Scope是定义Spring如何创建bean的实例的。 在创建bean的时候可以带上scope属性，scope有下面几种类型。 Singleton 这也是Spring默认的scope，表示Spring容器只创建一个bean的实例，Spring在创建第一次后会缓存起来，之后不再创建，就是设计模式中的单例模式

177. Spring cache
https://www.cnblogs.com/coding-one/p/12401630.html
Spring Cache uses the parameters of the method as key and the return value as a value in the cache. When the method is called the first time, Spring will check if the value with the given key is in the cache. It will not be the case, and the method itself will be executed.

The idea of the @Cacheable annotation is that you use it to mark the method return values that will be stored in the cache. The @Cacheable annotation can be applied either at method or type level. When applied at method level, then the annotated method's return value is cached.

178. Consume web service in Java
我不理解我不明白

179. Controller vs. RestController, what did you put in a controller annotations？
Follow up: why did you put some keywords like add or get in your url, is it ok to put id or quantity in the url? How about username or password?

@Controller is used to mark classes as Spring MVC Controller.
@RestController annotation is a special controller used in RESTful Web services, and it's the combination of @Controller and @ResponseBody annotation. It is a specialized version of @Component annotation.

180. Bean scope and explain the request scope
Bean scope see 176.
In request scope, a bean is defined to an HTTP request whereas in session scope, it is scoped to an HTTP session. So for an instance, if the bean scope is request and,

a user makes more than one request for a web page in his user session, then on every request a new bean would be created

182. What is Dependency Injection?
Dependency Injection is a fundamental aspect of the Spring framework, through which the Spring container “injects” objects into other objects or “dependencies”. Simply put, this allows for loose coupling of components and moves the responsibility of managing components onto the container

183. Different ways of DI
https://www.tutorialspoint.com/what-are-the-different-ways-to-implement-dependency-injection-and-their-advantages-in-chash

There are four types of DI:

- Constructor Injection.
- Setter Injection.
- Interface-based injection.
- Service Locator Injection.

Use Setter injection when a number of dependencies are more or you need readability.
Use Constructor Injection when Object must be created with all of its dependency.
What kind of injection is used when the dependency is used? **Setter based**

Setter based Injection vs. Constructor based
If we use both constructor and setter injection, IOC container will use the setter injection. Changes: We can easily change the value by setter injection. It doesn't create a new bean instance always like constructor. So setter injection is flexible than constructor injection.

184. Explain the procedure of how you implement a web service

1. About Creating Web Service References.
2. Creating a New Application.
3. Specifying an Application Proxy Server Address.
4. Creating a Web Service Reference from a WSDL.
5. Create a Form and Report.
6. Creating a Web Service Reference Manually.
7. Test the Web Service.

185. List some status codes you encountered
https://umbraco.com/knowledge-base/http-status-codes/

404 Not found
202 Accepted

186. Difference between @Component and @Bean
@Component is a class-level annotation, but @Bean is at the method level, so @Component is only an option when a class's source code is editable.
@Bean can always be used, but it's more verbose.
@Component is compatible with Spring's auto-detection, but @Bean requires manual class instantiation.

If we see component class like @Controller, @service, @repository will be scan automatically by the spring framework using the component scan.
@Bean on the other hand can only be used to explicitly declare a single bean in a configuration class.
https://www.jianshu.com/p/32a72bb38487

187. RestTemplate vs. Kafka
RestTemplate is used for synchronous call while Kafka is used for A-synchronous call

189. When there is circular dependency, use which type of Injection?
A circular dependency occurs when two classes depend on each other. For example, class A needs class B, and class B also needs class A. Circular dependencies can arise in Nest between modules and between providers. While circular dependencies should be avoided where possible, you can't always do so.
Constructor injection is used

190. Explain DispatcherServlet in detail
The DispatcherServlet is the front controller in Spring web applications. It's used to create web applications and REST services in Spring MVC. In a traditional Spring web application, this servlet is defined in the web. xml file.

What is DispatcherServlet How does it work?

DispatcherServlet handles an incoming HttpRequest, delegates the request, and processes that request according to the configured HandlerAdapter interfaces that have been implemented within the Spring application along with accompanying annotations specifying handlers, controller endpoints, and response objects.

191. What is race condition?

A race condition occurs when two or more threads access shared data and they try to change it at the same time. Thus, race conditions can cause runtime errors or unexpected outcomes.

192. Why do you use Spring Boot? / Benefits of Spring Boot

- Spring Framework can be employed on all architectural layers used in the development of web applications;
- Uses the very lightweight POJO model when writing classes;
- Allows you to freely link modules and easily test them;
- Supports declarative programming;
- Eliminates the need to independently create factory and singleton classes;
- Supports various configuration methods;
- Provides middleware-level service.

https://bambooagile.eu/insights/pros-and-cons-of-using-spring-boot/

193. Different types of request mapping methods
RequestMapping annotation is used to map web requests onto specific handler classes and/or handler methods. @RequestMapping can be applied to the controller class as well as methods. Today we will look into various usage of this annotation with example and other annotations @PathVariable and @RequestParam.

1. @GetMapping.
2. @PostMapping.
3. @PutMapping.
4. @DeleteMapping.
5. @PatchMapping.

194. SOAP vs. REST
SOAP is XML based, heavy weighted and carry status while REST is JSON based, light weighted and stateless.

195. ApplicationContext vs. BeanFactory
The ApplicationContext comes with advanced features, including several that are geared towards enterprise applications, while the BeanFactory comes with only basic features. Therefore, it's generally recommended to use the ApplicationContext, and we should use BeanFactory only when memory consumption is critical.
https://www.baeldung.com/spring-beanfactory-vs-applicationcontext

196. Talk about REST api / How do you deploy a REST api project? (AWS)

1. Sign in to the API Gateway console at https://console.aws.amazon.com/apigateway.
2. In the APIs navigation pane, choose the API you want to deploy.
3. In the Resources navigation pane, choose Actions.
4. From the Actions drop-down menu, choose Deploy API.
5. In the Deploy API dialog, choose an entry from the Deployment stage dropdown list.
6. If you choose [New Stage], enter a name in Stage name and optionally provide a description for the stage and deployment in Stage description and Deployment description. If you choose an existing stage, you might want to provide a description of the new deployment in Deployment description.
7. Choose Deploy to deploy the API to the specified stage with default stage settings.

197. How does JWT work?
JWT, or JSON Web Token, is an open standard used to share security information between two parties — a client and a server. Each JWT contains encoded JSON objects, including a set of claims. JWTs are signed using a cryptographic algorithm to ensure that the claims cannot be altered after the token is issued.

199. What is front controller?
Front Controller - In Spring Web MVC, the DispatcherServlet class works as the front controller. It is responsible to manage the flow of the Spring MVC application.

200. How do you implement transaction?
1. Locate the record to be updated from secondary storage
2. Transfer the block disk into the memory buffer
3. Make the update to tuple in the buffer buffer
4. Write the modified block back out to disk
5. Make an entry to a log

202. When designing a public API, what needs to be considered?

203. How to monitor the status of the application? / What dependency is used when monitoring the application?
The **Spring Cloud** Data Flow server is a Spring Boot application that includes the Actuator library, which adds several production ready features to help you monitor and manage your application.

204. What is DevTools?
DevTools stands for Developer Tool. The aim of the module is to try and improve the development time while working with the Spring Boot application. Spring Boot DevTools pick up the changes and restart the application. We can implement the DevTools in our project by adding the following dependency in the pom.

206. Authentication vs. Authorization
Authentication and authorization are two vital information security processes that administrators use to protect systems and information. Authentication verifies the identity of a user or service, and authorization determines their access rights.

207. What is the security part used for?
Spring Security is the primary choice for implementing application-level security in Spring applications. Generally, its purpose is to offer you a highly customizable way of implementing authentication, authorization, and protection against common attacks.

208. How does service communicate with each other?
Each service instance is typically a process. Therefore, services must interact using an inter-process communication protocol such as HTTP, AMQP, or a binary protocol like TCP, depending on the nature of each service.

209. Talk about MVC
The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application.

210. REST vs. RESTful
REST is the set of constraints. RESTful refers to an API adhering to those constraints.

212. Singleton scope
Description. singleton. Scopes a single bean definition to a single object instance per Spring IoC container. prototype. Scopes a single bean definition to any number of object instances.

214. How to inject bean when there are two beans with same type?
The default autowiring is by type, not by name, so when there is more than one bean of the same type, you have to use the @Qualifier annotation.

216. If define an object in the controller, and then other request update the value of the object, then which value will be fetched by a new request?

217. How to start a Spring Boot project?

Step 1: Open the Spring Initializr https://start.spring.io/.

Step 2: Select the Spring Boot version 2.2. ...

Step 3: Provide the Group name. ...

Step 4: Provide the Artifact. ...

Step 5: Add the Spring Web dependency.

Step 6: Click on the Generate button.

218. Command to run a spring boot application
java -jar target/
mvn spring-boot:run
gradle bootRun

https://www.appsdeveloperblog.com/run-spring-boot-app-from-a-command-line/

219. How to let Spring create instance for you?
Spring Boot @Autowired creating instances on a runtime

220. How to make sure the security of user request?
Harden authentication:
Spring Security simplifies authentication and helps you make it more secure. By default, it includes a form-based login page which verifies username and password provided by the user against the application users database, and then keeps track of the user authentication during their session.

221. How to make sure of different user security?

224. How to record logs using AOP?
1. Maven Dependency. To add AOP support, we must include spring-boot-starter-aop dependency in application. ...
2. Performance Logging Aspect. Create a spring bean and use annotation @Aspect on class to make it AOP aspect. ...
3. Classes and intercepted methods. ...
4. Spring boot performance logging example.

https://howtodoinjava.com/spring-boot2/logging/performance-logging-aspectj-aop/

225. How to handle controller exceptions?
You can add extra (@ExceptionHandler) methods to any controller to specifically handle exceptions thrown by request handling (@RequestMapping) methods in the same controller. Such methods can: Handle exceptions without the @ResponseStatus annotation (typically predefined exceptions that you didn't write)
226. Spring validation?

Spring features a Validator interface that you can use to validate objects. The Validator interface works using an Errors object so that while validating, validators can report validation failures to the Errors object.

227. What does @SpringBootApplication mean?

@SpringBootApplication annotation is used to mark a configuration class that declares one or more @Bean methods and also triggers auto-configuration and component scanning.

228. How do you create a controller?

In Spring Boot, the controller class is responsible for processing incoming REST API requests, preparing a model, and returning the view to be rendered as a response. The controller classes in Spring are annotated either by the @Controller or the @RestController annotation.

229. What is routing?

Routing is the process of selecting a path for traffic in a network or between or across multiple networks. Broadly, routing is performed in many types of networks, including circuit-switched networks, such as the public switched telephone network (PSTN), and computer networks, such as the Internet.

230. How do you create a rest controller?

1. Update maven dependencies. .
2. Add ContentNegotiatingViewResolver. Update bean configuration file for view resolvers and add ContentNegotiatingViewResolver.
3. Add JAXB annotations in model classes.
4. Create REST Controller.
5. Demo for spring rest example

231. What is path variable?

@PathVariable is a Spring annotation which indicates that a method parameter should be bound to a URI template variable.

232. What is service discovery?

Advertisements. Eureka Server is an application that holds the information about all client-service applications. Every Micro service will register into the Eureka server and Eureka server knows all the client applications running on each port and IP address. Eureka Server is also known as Discovery Server.

233. For Spring Boot, how do I use another server such as WebLogic instead of the embedded Tomcat server?

https://stackoverflow.com/questions/63054114/how-can-i-change-server-in-spring-boot-application

234. Constructor based injection vs. Setter based injection, when to use which?

Use Setter injection when a number of dependencies are more or you need readability.
Use Constructor Injection when Object must be created with all of its dependency.

235. How to implement security?

236. How to use docker in the Spring Boot?

1. Create a Spring Boot Application.
2. Create Dockerfile.
3. Build executable jar file.
4. Build Docker image.
5. Run Docker container using the image built.
6. Test.

https://medium.com/geekculture/docker-basics-and-easy-steps-to-dockerize-spring-boot-application-17608a65f657

238. Explain MVC pattern

MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

Model - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.

View - View represents the visualization of the data that model contains.

Controller - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

239. What will happen after click the URL?

As we know, when we hit any URL or you can say domain name, then that website gets opened with its content. A server (a trained computer) serves it. We also know that every computer has an IP address which is used for communication over the internet. It is an address as its self explaining 'IP address'.

- We click the URL
- The browser resolves that URL into an IP Address via DNS.
- A TCP connection is initiated between the client and server
- The connection is formalized and Requests & Responses can be sent.

https://levelup.gitconnected.com/clicking-a-link-what-happens-next-d3360553735a

240. Ways to connect to database

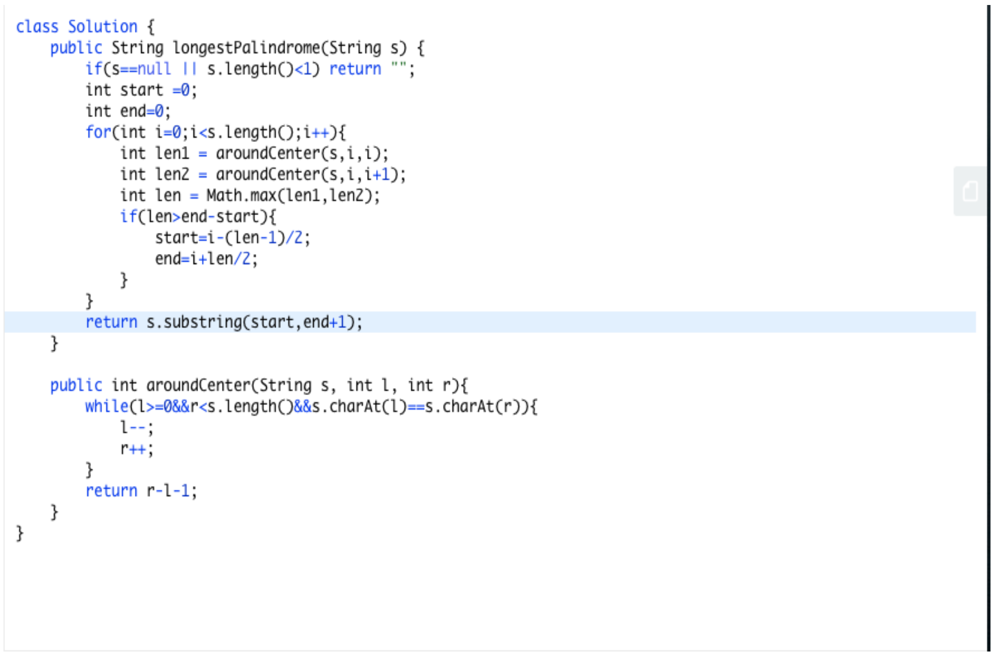
1. Step 1 - Add dependency for your database connector to pom. xml. ...
2. Step 2 - Remove H2 Dependency from pom.xml. Or atleast make its scope as test. ...
3. Step 3 - Setup your My SQL Database. ...
4. Step 4 - Configure your connection to Your Database
5.

241. Can we make a class like Student singleton?

No! 只有无状态的class才能被定义为单例模式!!!!!!
student中有name age等能被修改的变量 所以状态会改变 是有状态的class!!!

1

Leetcode 5 ：Longest palindrome substring by Yuehao

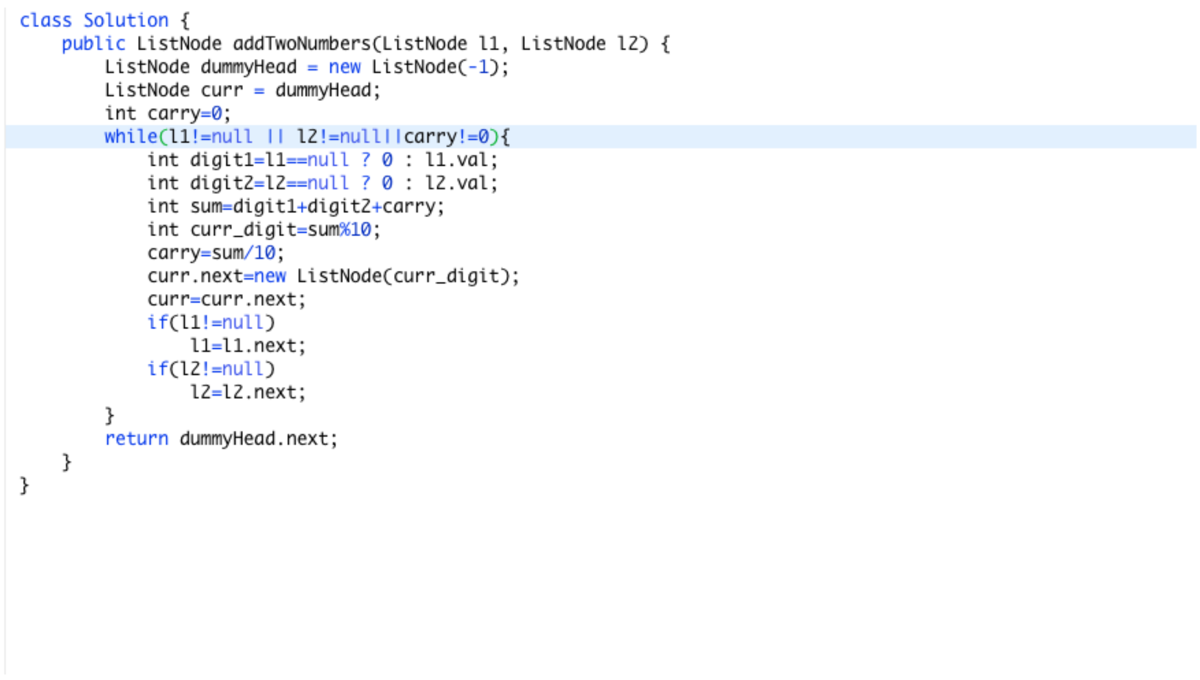


```java
class Solution {
    public String longestPalindrome(String s) {
        if(s==null || s.length()<1) return "";
        int start =0;
        int end=0;
        for(int i=0;i<s.length();i++){
            int len1 = aroundCenter(s,i,i);
            int len2 = aroundCenter(s,i,i+1);
            int len = Math.max(len1,len2);
            if(len>end-start){
                start=i-(len-1)/2;
                end=i+len/2;
            }
        }
        return s.substring(start,end+1);
    }

    public int aroundCenter(String s, int l, int r){
        while(l>=0&&r<s.length()&&s.charAt(l)==s.charAt(r)){
            l--;
            r++;
        }
        return r-l-1;
    }
}
```

2

Leetcode2: Add two numbers by Yuehao



```java
class Solution {
    public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
        ListNode dummyHead = new ListNode(-1);
        ListNode curr = dummyHead;
        int carry=0;
        while(l1!=null || l2!=null||carry!=0){
            int digit1=l1==null ? 0 : l1.val;
            int digit2=l2==null ? 0 : l2.val;
            int sum=digit1+digit2+carry;
            int curr_digit=sum%10;
            carry=sum/10;
            curr.next=new ListNode(curr_digit);
            curr=curr.next;
            if(l1!=null)
                l1=l1.next;
            if(l2!=null)
                l2=l2.next;
        }
        return dummyHead.next;
    }
}
```

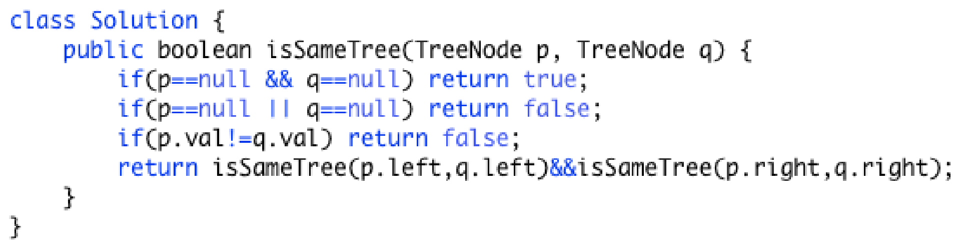
3

Use two threads to print out an Array, one prints odds and the other prints evens.

https://www.geeksforgeeks.org/print-even-and-odd-numbers-in-increasing-order-using-two-threads-in-java/

4

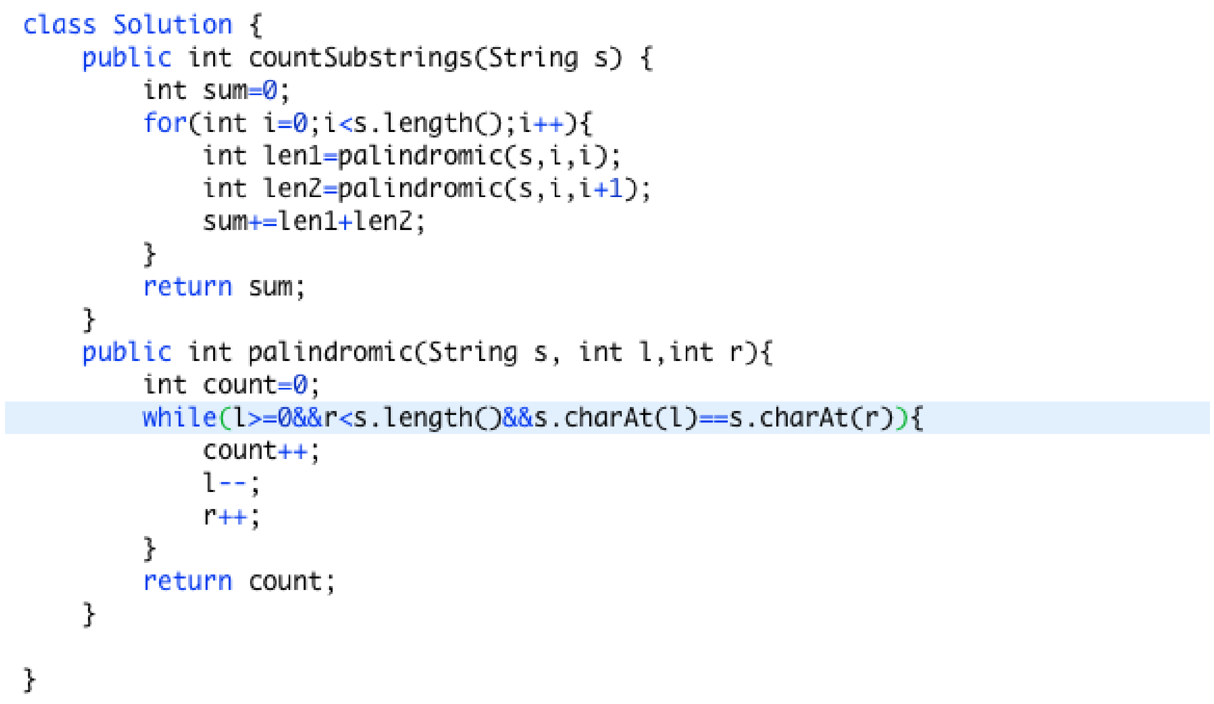
Check if two binary search tree are identical
Leetcode100: Same tree by Yuehao



```java
class Solution {
    public boolean isSameTree(TreeNode p, TreeNode q) {
        if(p==null && q==null) return true;
        if(p==null || q==null) return false;
        if(p.val!=q.val) return false;
        return isSameTree(p.left,q.left)&&isSameTree(p.right,q.right);
    }
}
```

5

Leetcode 647: Sub-palindrome by Yuehao

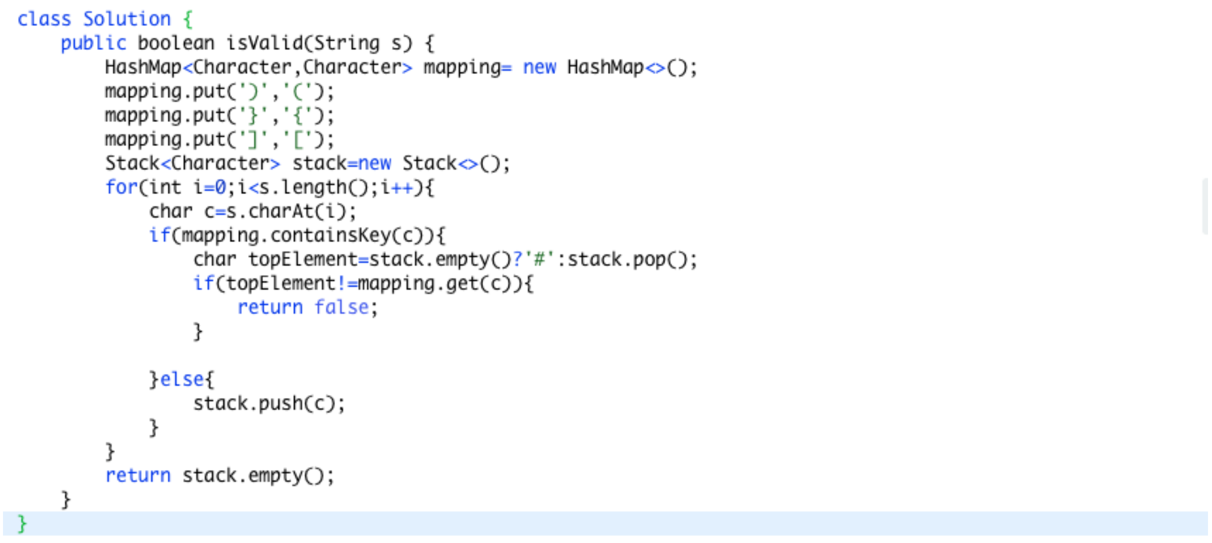


```java
class Solution {
    public int countSubstrings(String s) {
        int sum=0;
        for(int i=0;i<s.length();i++){
            int len1=palindromic(s,i,i);
            int len2=palindromic(s,i,i+1);
            sum+=len1+len2;
        }
        return sum;
    }
    public int palindromic(String s, int l,int r){
        int count=0;
        while(l>=0&&r<s.length()&&s.charAt(l)==s.charAt(r)){
            count++;
            l--;
            r++;
        }
        return count;
    }

}
```

6

Leetcode 20: Valid Parentheses by Yuehao

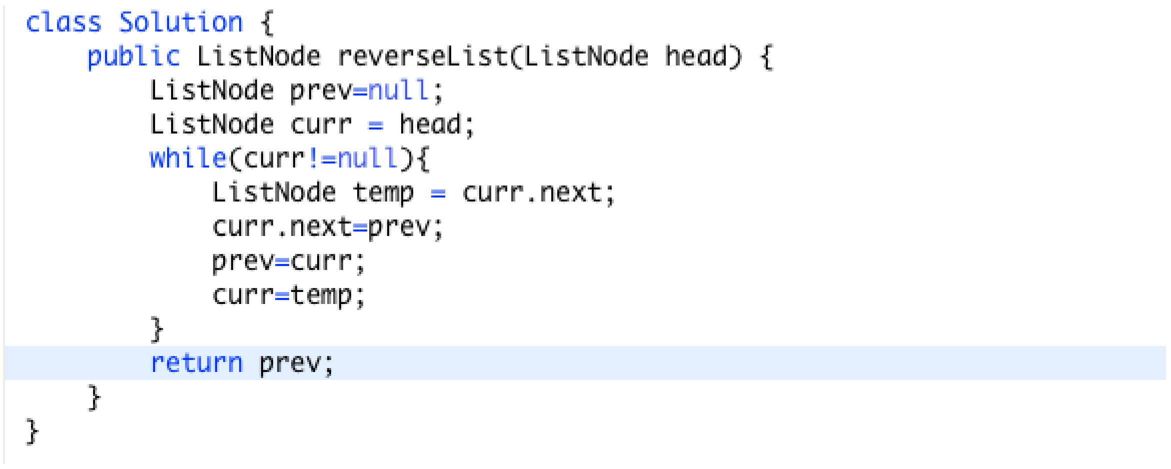


```java
class Solution {
    public boolean isValid(String s) {
        HashMap<Character,Character> mapping= new HashMap<>();
        mapping.put(')','(');
        mapping.put('}','{');
        mapping.put(']','[');
        Stack<Character> stack=new Stack<>();
        for(int i=0;i<s.length();i++){
            char c=s.charAt(i);
            if(mapping.containsKey(c)){
                char topElement=stack.empty()?'#':stack.pop();
                if(topElement!=mapping.get(c)){
                    return false;
                }

            }else{
                stack.push(c);
            }
        }
        return stack.empty();
    }
}
```

7

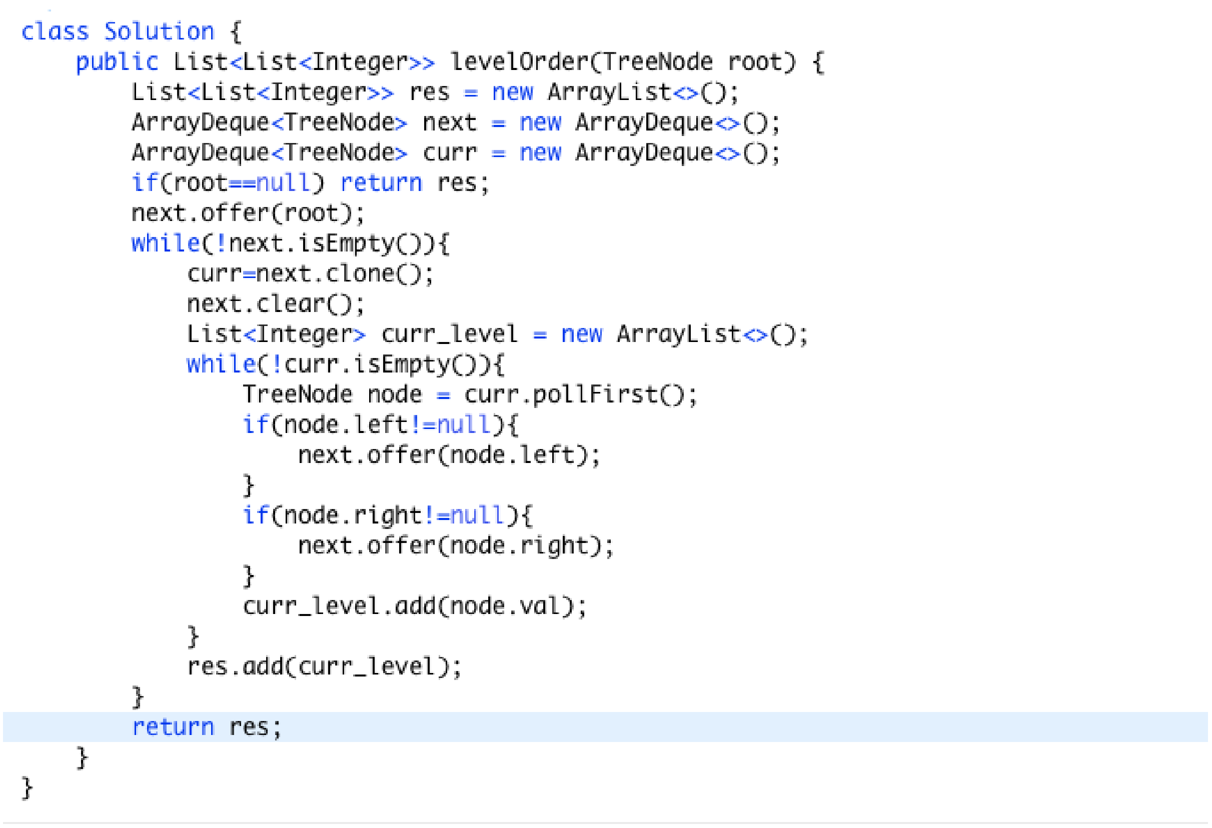
Leetcode 206: Reverse Linked List   by Yuehao



```java
class Solution {
    public ListNode reverseList(ListNode head) {
        ListNode prev=null;
        ListNode curr = head;
        while(curr!=null){
            ListNode temp = curr.next;
            curr.next=prev;
            prev=curr;
            curr=temp;
        }
        return prev;
    }
}
```

8

Leetcode 102:Binary Tree Level Order Traversal   by Yuehao



```java
class Solution {
    public List<List<Integer>> levelOrder(TreeNode root) {
        List<List<Integer>> res = new ArrayList<>();
        ArrayDeque<TreeNode> next = new ArrayDeque<>();
        ArrayDeque<TreeNode> curr = new ArrayDeque<>();
        if(root==null) return res;
        next.offer(root);
        while(!next.isEmpty()){
            curr=next.clone();
            next.clear();
            List<Integer> curr_level = new ArrayList<>();
            while(!curr.isEmpty()){
                TreeNode node = curr.pollFirst();
                if(node.left!=null){
                    next.offer(node.left);
                }
                if(node.right!=null){
                    next.offer(node.right);
                }
                curr_level.add(node.val);
            }
            res.add(curr_level);
        }
        return res;
    }
}
```